MageAdd: Real-Time Interaction Simulation for Scene Synthesis

Shao-Kui Zhang
zhangsk18@mails.tsinghua.edu.cn
Tsinghua University
China

Yi-Xiao Li
liyixiao20@mails.tsinghua.edu.cn
Tsinghua University
China

Yu He
hooyeeevan@tsinghua.edu.cn
Tsinghua University
China

Yong-Liang Yang
y.yang@cs.bath.ac.uk
University of Bath
United Kingdom

Song-Hai Zhang*
shz@tsinghua.edu.cn
Tsinghua University & BNRist
China

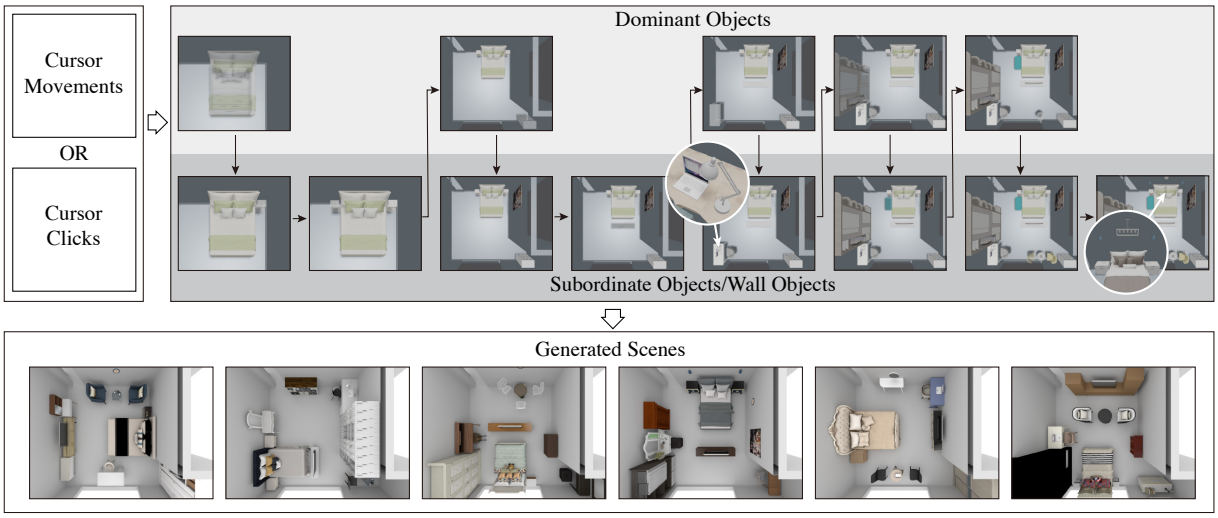


Figure 1: We present an interactive framework for synthesizing 3D scenes by iteratively inferring objects and their transformations based on cursor movements and clicks. Given a cursor movement at any moment, our framework automatically selects, translates and rotates an object plausibly into the scene. Mouse click would end an iteration and refresh priors, which introduces more potential objects and constraints. While the cursor moves, we achieve real-time arrangement of objects with proper transformations.

ABSTRACT

While recent researches on computational 3D scene synthesis have achieved impressive results, automatically synthesized scenes do not guarantee satisfaction of end users. On the other hand, manual scene modelling can always ensure high quality, but requires a cumbersome trial-and-error process. In this paper, we bridge the above gap by presenting a data-driven 3D scene synthesis framework that can intelligently infer objects to the scene by incorporating and simulating user preferences with minimum input. While the cursor is moved and clicked in the scene, our framework automatically selects and transforms suitable objects into scenes in real time. This is based on priors learnt from the dataset for placing different types of objects, and updated according to the current scene context. Through extensive experiments we demonstrate that our framework outperforms the state-of-the-art on result aesthetics, and enables effective and efficient user interactions.[1]

*Corresponding Author, with Tsinghua University and Beijing National Research Center for Information Science and Technology (BNRist).

CCS CONCEPTS

• **Computing methodologies** → *Graphics systems and interfaces.*

KEYWORDS

3D Indoor Scene Synthesis, Spatial Inference, User Interaction.

ACM Reference Format:
Shao-Kui Zhang, Yi-Xiao Li, Yu He, Yong-Liang Yang, and Song-Hai Zhang. 2021. MageAdd: Real-Time Interaction Simulation for Scene Synthesis. In *Proceedings of the 29th ACM International Conference on Multimedia (MM*

[1]Code is available at https://github.com/Shao-Kui/3DScenePlatform#mageadd.

'21), October 20–24, 2021, Virtual Event, China. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475194

1 INTRODUCTION

3D scene synthesis is an active area for both academia and industry, and benefits various important applications. First, virtual indoor scene generation is an essential topic for environment design [1, 5, 13, 23]. Video games also have increasing demands for indoor scenes to enhance user experiences [6]. Well-configured virtual scenes can be used to generate synthetic datasets for computer vision applications [12]. Virtual reality applications also require realistic scenes as virtual training examples [15].

Synthesizing plausible 3D indoor scenes has been investigated in the last years with various input, techniques, and applications [37]. A general workflow adopted by most literature is selecting a set of appropriate objects and placing them plausibly in a given room [11, 16], whereas few works focus on the arrangement of objects by assuming a selection of objects are given [29, 34]. With the development of recent works, results with better plausibility and aesthetics are achieved for automatic scene generations [27, 39].

Despite the progress on computational 3D scene synthesis, fully automatic results are not guaranteed for user satisfaction. End users still have their own preferences to select and arrange indoor objects. In practice, interior designers still need to prepare several designs for end users to select from and to propose further adjustments[2]. Consequently, it makes more sense to incorporate user preferences into scene synthesis. However, we can not expect novice users to manually select and arrange objects for the whole process, which is costly even for professionals. Thus how to effectively and efficiently suggest and transform objects while minimizing the efforts of user interaction is the key problem to solve. As such, our aim is to achieve real-time interactions where object selections and arrangements [37] are incorporated with minimum user inputs of mouse movements. In other words, when the cursor moves, objects should be well selected and placed along the way. This allows the user to easily explore the scene arrangements with potential objects, such that the result can meet the user expectation as much as possible.

In this paper, we present an interaction-based framework for iteratively synthesizing 3D scenes as shown in Figure 1. Given an empty room or a room with some existing objects, the minimum user input is to just move the cursor in the scene. While moving the cursor, our framework automatically selects and places suitable object into the scene based on the current scene context and the scene arrangement priors learnt from a 3D scene dataset. More specifically, the prior that best matches the current cursor location in the scene is used to add object to the scene. The position, orientation, and size of the object are also optimized to adapt to the current scene context. The user has extra freedom to swap objects guided by other strong priors. After the user confirms a suggested object, the priors will be updated immediately according to the new scene context. The above interaction iteratively performs until satisfactory result is achieved. Note that when arranging objects in the scene, we classify objects as dominant objects (e.g., wardrobe, bed, dinning table), subordinate objects (e.g., dinner chair, nightstand), and wall objects/decorations (e.g., painting, wall lamp). Each class has a specific way of learning priors and inferring objects. The intuition is that humans placing objects follows an order of importance [22, 26, 40], i.e., *organizational flows* in interior design [22]. For example, in a master bedroom, a double bed is placed first and the accompanied nightstand(s) are placed subsequently. Some objects such as cabinets are also assembled at the early stage of the interior design to lay out room space [18, 26].

To evaluate our work, we first test the ease of interactions by conducting a user study. It shows that our work significantly reduces the interaction time compared with the existing solutions. Second, we perform both qualitative and quantitative comparisons between our work and the existing baselines on automatically generated scenes, demonstrating higher plausibility and aesthetics. Finally, we exhibit the efficiency of our framework through tests on various devices with different hardware configurations.

Overall our work makes the following major contributions:

- We present a framework for synthesizing user-satisfied scenes by incorporating user preferences through convenient cursor movements.
- We simplifies traditional user interactions of selecting and arranging objects into suggesting objects based on cursor movements through a data-driven approach.
- We learn specific priors for different objects, enabling both real-time and plausible inferences.

2 RELATED WORKS

The most common input for 3D indoor scene synthesis is an initial 3D scene with boundaries, e.g., grounds and walls [16, 21, 24, 27, 33, 34, 38, 39]. Xu et al. [31] generate 3D scenes guided by hand-drawn sketches. Human languages are also decoded as graphs for scene generations [2, 20]. RGB-D scans are also used for the physical guidance of scene synthesis [3, 4, 8]. Luo et al. [19] even generate scenes with RGB images. Several works synthesize scenes based on other reference scenes [7, 30]. Our method is initialized directly with an empty scene or optionally a scene with several existing objects.

In terms of concrete technologies, stochastic optimization strategies such as Markov Chain Monte Carlo (MCMC) is often adopted if learnt priors are non-differentiable expressions [16, 24, 33]. In contrast, gradient descent is used if priors are differentiable [7, 21, 31]. Graph techniques are also frequently used to construct intuitive relations between objects [2, 14, 24], where scenes are generated according to a constructed graph with scene attributes stored on graph vertices and edges. Recently, the plausibility is even improved by leveraging the development of neural networks [27, 28, 41, 42], or adopting the concept of computational geometry [39]. Although several existing works do incorporate human activities when synthesizing scenes [8, 11], the "agent behavior" needs to be further considered. Yet, researches have not tried synthesizing 3D scenes by simulating end user interactions, where an end user typically manipulates a given scene by selecting preferable objects and placing them properly into the scene. We show that better plausibility can be achieved by simulating user interactivity for scene synthesis.

Synthesizing scenes by incorporating user interactions is much less explored. [25] selects objects given mouse clicks, while we infer both object selections and transformations during mouse

[2]https://planner5d.com/

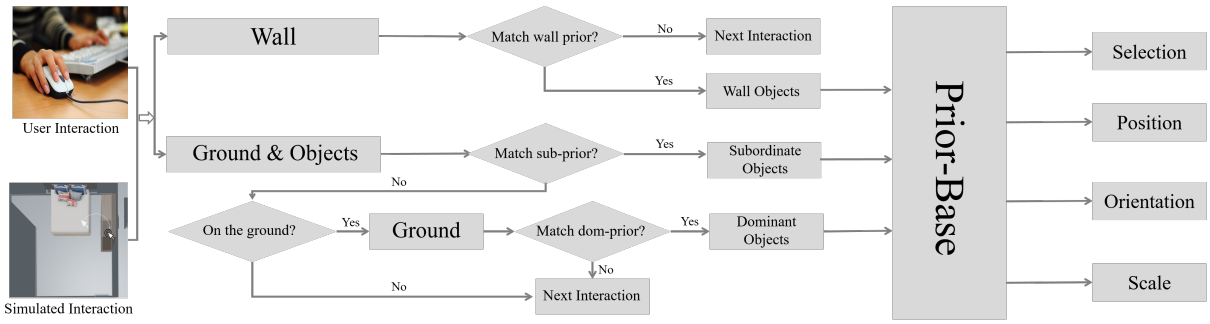


Figure 2: The general workflow of a single movement. After obtaining a movement of the cursor, we first compute intersected positions based on ray casting from the perspective camera to directions of the cursor. If the intersection falls on walls and matches one or more wall-priors, a wall instance is selected and transformed based on the best matched prior. Otherwise, it proceeds to the next iteration with regard to a new movement. If the intersection does not fall on walls, it first tries matching sub-priors. If matched, a subordinate object is inserted similarly based on priors. If no sub-prior is matched and the intersection is on the ground, we finally try inserting dominant objects.

movement. [35, 36] pop up detailed and small objects to enrich existing scenes, while our method can generate scenes with objects at different levels from scratch. [32] optimizes scenes based on example scenes using MCMC under user-specified constraints, thus is hard to achieve the real-time performance.

3 OVERVIEW

Our framework iteratively inserts objects into the given scene. Each iteration corresponds to a single movement of the cursor. While the cursor moves, a suggested object may emerge in the current scene according to the *unprojected direction* indicated by the cursor. This direction is computed by a ray casted from the camera center (eye point) to the cursor (image point), suppose the cursor moves along a rendered image plane lying between the camera center and the scene. We always take the first intersection point between the ray and the 3D scene, which is denoted as $\mathbf{\Lambda} = (x_{\mathbf{\Lambda}}, y_{\mathbf{\Lambda}}, z_{\mathbf{\Lambda}})$ for object insertion.

Figure 2 shows the process of a single iteration. Given an intersection point $\mathbf{\Lambda}$, our framework first checks whether $\mathbf{\Lambda}$ is casted on a wall or not. If so, it attempts to suggest wall objects based on wall priors (Section 6). Otherwise, it proceeds to suggest subordinate objects based on sub-priors (Section 5). If no sub-prior is applicable to $\mathbf{\Lambda}$ and $\mathbf{\Lambda}$ is on the ground, it seeks dominant objects based on dom-priors (Section 4). If none of the above priors is feasible for $\mathbf{\Lambda}$, the next iteration is required with a new $\mathbf{\Lambda}$ by moving the cursor. In contrast, if several priors are matched, we select the best one among all priors in a prior base, where the object selection and transformation are determined w.r.t. $\mathbf{\Lambda}$.

The prior base contains a set of priors related to potential objects that can be inserted to the current scene. Each prior has an index to a specific object. It also contains an object-scene context and how to arrange the object in the context. For example, given a dinning table in a room, if $\mathbf{\Lambda}$ is adjacent to it, $\mathbf{\Lambda}$ would possibly match a prior indexing to dinning chairs. The ways to learn and utilize priors for object insertion are distinct for dominant objects, subordinate objects, and wall objects, which will be presented respectively in the following three sections. Unlike fitting models and re-sampling them, we directly take samples as priors, where samples are extracted and processed directly from the dataset. An observation is that we can not hypothesize the learnt models (e.g., Gaussian mixed model, neural networks) , while data samples sufficiently present prior knowledge of scene arrangement (see Section 7).

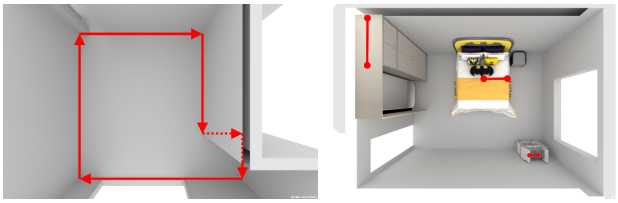
The content and structure of the prior base are fixed unless new object is inserted to the scene. For example, inserting a double bed in a room subsequently triggers the addition of new priors of nightstands to the prior base. The construction of the prior base is according to the current objects in the scene. With the help of the prior base, our framework automatically suggests different objects according to different intersection point $\mathbf{\Lambda}$ while the cursor moves.

Note that additional features are incorporated to enable personalization (see Section C of the supplementary), but our main aim is to give the user easy control of the scene synthesis process in real time, where the key challenge is how to organize different types of objects according to room type, size, object/wall location, etc. This motivates us to learn adaptive priors which can efficiently guide the synthesis process. Based on the automatic and real-time results (either final or intermediate), the user can also swap/filter object.

4 DOMINANT OBJECTS

Following interior guidelines [18, 22, 26, 40], we introduce the "Room Depth Model" (RDM) for selecting and transforming a dominant object from R candidates $\{o^r | r \in [1, R]\}$, if $\mathbf{\Lambda}$ matches no wall prior or subordinate prior. Under the assumption of RDM, a room with n sides is abstracted as an n-gon composed by a set of (clockwise) edge vectors $\{\mathbf{v}^i | i = 1, 2, ..., n\}$ as shown in Figure 3a.

Firstly, for each mouse movement iteration, we find the nearest wall $v_{(1)}$ and the second nearest wall $v_{(2)}$ of $\mathbf{\Lambda}$ and the respective distances $d_{(1)}$ and $d_{(2)}$ of $\mathbf{\Lambda}$ to the two walls (see Section B of the supplementary document for details). Secondly, with $d_{(1)}$ and $d_{(2)}$, we compare $d_{(1)}$ with priors extracted from the dataset. Thus, we next introduce our way to extract priors for RDM. For the r-th *dominant object* of its j-th *occurrence* in a room, we record its ID (name) $p^{r,j}_{id}$, its nearest-wall distance $p^{r,j}_{dis}$, the difference of the orientation of the object and the nearest wall $p^{r,j}_{ori}$, its scale $p^{r,j}_{scale}$,



(a) Representing shapes of rooms. (b) The "p_{len}" of three objects.

Figure 3: Illustration of the RDM. 3a: the shape of each room is abstracted as a polygon with n sides, where edges are vectorized and connected clock-wisely. 3b: half the length of objects against nearest walls.

and half the length $p^{r,j}_{len}$ of it against its nearest wall as shown in Figure 3b. The orientation of a wall is represented by the orientation of its inward normal. Therefore, for each candidate dominant object, we can generate one prior $\mathcal{P}^{r,j} = \{p^{r,j}_{id}, p^{r,j}_{dis}, p^{r,j}_{ori}, p^{r,j}_{scale}, p^{r,j}_{len}\}$ per occurrence and each dominant object eventually has a prior set of RDM, that is $\mathcal{D}^r = \{\mathcal{P}^{r,1}, \mathcal{P}^{r,2}, \ldots, \mathcal{P}^{r,\epsilon(r)}\}$. Note that one object may occur twice or more in a single room and $\epsilon(\cdot)$ denotes the number of total occurrences.

With a room and a set of R candidate dominant objects, $\{o^r | r \in [1, R]\}$, $\{\mathcal{D}^1, \mathcal{D}^2 \ldots, \mathcal{D}^R\}$ is pre-computed and loaded, so we can always find a $\mathcal{P}^{\hat{r},\hat{j}}$ satisfying the following equation,

$$(\hat{r}, \hat{j}) = \underset{r\in[1,R], j\in[1,\epsilon(r)]}{\arg\min} \varphi(p^{r,j}_{dis}, p^{r,j}_{len}, d_{(1)}, d_{(2)}), \tag{1}$$

$$\varphi(p_{dis}, p_{len}, d_{(1)}, d_{(2)}) = \begin{cases} |p_{dis} - d_{(1)}|, & \text{if } d_{(2)} \geq p_{len} \\ +\infty, & \text{otherwise} \end{cases}. \tag{2}$$

Seeking $\hat{r}$ and $\hat{j}$ is equivalent to seeking a $\mathcal{P}^{r,j}$ among the RDM prior sets of the R objects, where j is less than the prior set size $\epsilon(r)$ of object o^r. $\mathcal{P}$ should indeed have the lowest value of $\varphi(\cdot)$, which takes four values: $p_{dis}, p_{len}, d_{(1)}$ and $d_{(2)}$. The former two are nearest distance and against-wall length from datasets. The latter two are calculated using Λ for each iteration. $\varphi(\cdot)$ returns differences of p_{dis} and $d_{(1)}$, which are respectively nearest-wall distances from datasets and intersections in real-time, if $d_{(2)}$ is not less than p_{len}. Otherwise, it simply returns a "$+\infty$". In other words, we would try computing and searching a $\mathcal{P}^{r,j}$ with the closest nearest-wall distance $p^{r,j}_{dis}$ to the real-time nearest-wall distance $d_{(1)}$. However, if its second nearest-wall distance $d_{(2)}$ exceeds $p^{r,j}_{len}$, which means the remaining extent is insufficient to accommodate an object with against-wall length $2 \times p^{r,j}_{len}$.

Subsequently, back to an intersection Λ of an iteration, if the best matched $\varphi(p^{\hat{r},\hat{j}}_{dis}, p^{\hat{r},\hat{j}}_{len}, d_{(1)}, d_{(2)})$ is greater than a threshold η_{dom} (see Section C in the supplementary document for discussions of interactive thresholds), $(\hat{r}, \hat{j})$ is considered being too far away from the priors and no dom-object is suggested. Otherwise, the selected instance from the recommended R objects is indexed by $p^{\hat{r},\hat{j}}_{id}$. The translation of the object is set as Λ. The orientation of the object is set as $\theta_{(1)} + p^{\hat{r},\hat{j}}_{ori}$, where $\theta_{(1)}$ follows the inward normal of $v_{(1)}$. The scale of the object follows $p^{\hat{r},\hat{j}}_{scale}$.

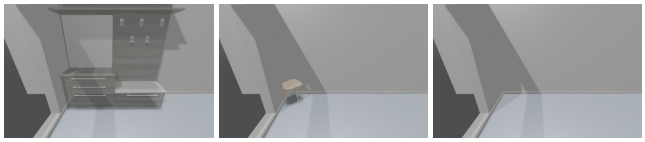


Figure 4: A scenario for illustrating Eqn. 1 and 2. With the intersection moving from right to left, the available second nearest-wall distance decrease. It first tries other less plausible instances with suitable sizes. As the distance even decreased, no instance fits the tiny area next to the corner.

Figure 4 shows a scenario of real-time calculations of $(\hat{r}, \hat{j})$. Our framework first suggests a wardrobe. When the intersection approaches the left, $d_{(2)}$ decreases and our framework resorts to another instance which is a corner/side table with suitable sizes. Finally, if the intersection is extremely close to the corner, no instance is acceptable any more. RDM is also robust to various room shapes. More qualitative results are shown in Figure 7 in the experiments.

In practice, it turns to be more efficient to select R candidate objects $\{o^r | r \in [1, R]\}$ in advance, based on the current scene contexts with previously selected objects. Although our framework is capable of recommending a set of candidate objects for each iteration, we still need an initial selection of objects for an empty room, which refers to the "cold booting" and is based on the And-Or Graph adapted by [24]. This "cold booting" is performed only once at the beginning, then we maintain a candidate object set by either adding or deleting object(s), depending on the process of the following iterations. For example, inserting a dom-object in a room results in deleting it from the candidate object set and adding related subordinate objects into the set. The initial room type is either given by the user or randomly selected by our framework. End users can also optionally specify dominant objects they prefer.

5 SUBORDINATE OBJECTS

Prior to inferring dominant objects, our framework tries matching subordinate priors first, where Λ may either be on the ground or elsewhere on the existing objects. Arranging sub-objects depends on the transient transformations of their dom-objects and the corresponding sub-prior sets, since different pairs of objects have distinct layout strategies as shown in Figure 5. Thus, similar to RDM, a set of subordinate priors $\mathcal{S}^{d,s} = \{Q^{d,s,1}, Q^{d,s,2}, ..., Q^{d,s,\varepsilon(d,s)}\}$ exist if

Figure 5: Example layout patterns between dominant objects and subordinate objects: Double Bed & Rug (Left), Double Bed & Nightstand (Middle), Dinning Table & Chair (Right). Colors denote orientations of sub-object following HSV color space where pure cyan is 0 degree.

a dominant object o^d is spatially related to a subordinate object o^s. $\varepsilon(d,s)$ equals to the number of co-occurrences of o^d and o^s in the prior set. Each $Q^{d,s,j}$ includes a plausible transformation $q_T^{d,s,j}$ of o^s w.r.t. o^d and the ID (name) $q_{id}^{d,s,j}$ of o^s.

We learn transformations (i.e., $q_T^{d,s,j}$) between dom-objects and sub-objects by adopting [38], which answers whether two objects are spatially related and what their layout patterns are as shown in Figure 5. A learnt transformation of a sub-object according to a dom-object includes a plausible translation and rotation w.r.t. Y-axis. Thus, the learnt transformation is assigned to $q_T^{d,s,j}$. An advantage of [38] is that it directly returns discrete samples as plausible transformations instead of hypothesizing concrete distributions , thus $q_T^{d,s,j}$ is robust to various layout patterns in practice.

When the cursor is associated with a dom-object, all related subordinate priors are loaded. Each $q_T^{d,s,j}$ is further transformed according to the dom-object, e.g., if a double bed is shrunk, sub-priors of a nightstand should be shrunk as well. Otherwise, an obvious gap will appear between the bed and the nightstand. Similarly, sub-objects are positioned and rotated following their dom-objects.

In the current context with R existing dom-objects, each dom-object o^d may be related to one or more sub-priors, resulting in $\mathcal{S}^{d,1} \cup \mathcal{S}^{d,2} \cup ... \cup \mathcal{S}^{d,\tau(d)}$ where $\tau(d)$ denotes the number of potential sub-objects related to o^d. Thus, we take the union of sub-priors of all dom-objects as $\mathcal{A}$. Note that any modification of dom-objects, such as transformations or deletion, results in adjustments of $\mathcal{A}$. Thus, we take the translation from $q_T^{d,s,j}$ as $q_{pos}^{d,s,j}$ and find the best matched prior $Q^{\hat{d},\hat{s},\hat{j}}$ by seeking a $q_{pos}^{d,s,j}$ nearest to $\mathbf{\Lambda}$ as shown in Eqn. 3:

$$(\hat{d},\hat{s},\hat{j}) = \underset{d\in[1,D],s\in[1,\tau(d)],j\in[1,\varepsilon(d,s)]}{\arg\min} ||q_{pos}^{d,s,j} - \mathbf{\Lambda}||_2. \quad (3)$$

If $||q_{pos}^{\hat{d},\hat{s},\hat{j}} - \mathbf{\Lambda}||_2$ is greater than a threshold η_{sub}, $(\hat{d},\hat{s},\hat{j})$ are considered too far away from $\mathcal{A}$ and no sub-object is suggested. If $\mathbf{\Lambda}$ is on the ground, we subsequently try inferring a dom-object. Otherwise, if η_{sub} is greater, the selected object are indexed by $q_{id}^{\hat{d},\hat{s},\hat{j}}$. The translation of the object is set as $(x_\Lambda, Y, z_\Lambda)$. The rotation of the object is set as $q_{ori}^{\hat{d},\hat{s},\hat{j}}$, which is the rotation extracted from $q_T^{\hat{d},\hat{s},\hat{j}}$. Y is the translation on the Y-axis of $q_{pos}^{\hat{d},\hat{s},\hat{j}}$ (the Y-axis is considered as "height" in this paper). For example, a rug is commonly placed beneath its dom-objects, e.g., double beds. However, rugs typically overlap with its dom-objects, and $\mathbf{\Lambda}$ is casted on dom-objects. Therefore, introducing Y alleviates the problem that objects are in different tiers [34].

6 WALL DECORATIONS

There are plenty of objects attached on walls instead of standing on the ground or other objects. Walls are commonly ignored in existing works, e.g., [27] trains convolutional neural networks for scene syntheses based on top-down orthographically rendered images. However, wall objects such as paintings are either rendered in tiny size or covered by other objects, which largely affects the quality of the results. However, wall objects are important components of home decoration. Few existing works do arrange wall objects, e.g., [39] recognizes wall objects as independent dom-objects, which makes sense since they are much more flexible to be placed compared with sub-objects. On the other hand, we argue that wall objects are often spatially related to dom or sub objects. For example, it is plausible to see wall lamps hung above nightstands. As such, in this section we present a method of arranging wall objects in coordination with objects on the ground. To our best knowledge, we are the first to specifically consider the layouts of wall objects.

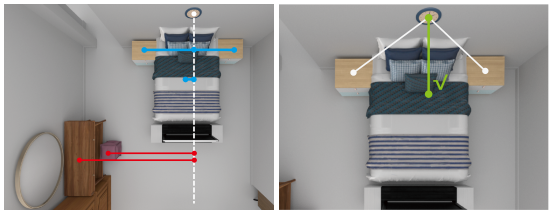


Figure 6: Finding a guide object. LEFT: Calculating λ of each object, given an intersection. A beam is shot from the intersection point toward the normal of the wall (a white dotted line). Thus, λ equals to the distance from objects to the beam (blue and red lines), where objects are discarded if their λ is too large (red). RIGHT: Selecting the guide object (green).

If $\mathbf{\Lambda}$ is casted on the wall area, we find its nearest wall. Subsequently, $\mathbf{b} = (x_\mathbf{b}, 0, z_\mathbf{b})$ is calculated as the inward normal vector of the casted wall, where its height on the Y-axis equals to 0. Next, we try finding a guide object among existing dom- and sub-objects. Given $\mathbf{b}$ and $\mathbf{\Lambda}$, a line $\psi(t)$ is formed as shown in Eqn. 4, where we shoot a "beam" from $\mathbf{\Lambda}$ to the direction of $\mathbf{b}$.

$$\psi(t) = \mathbf{\Lambda} + t \times \mathbf{b}, t \in (0, +\infty). \quad (4)$$

For each object o^i in the room, we take its translation on the XoZ plane as $\mathbf{w} = (x_i, y_\Lambda, z_i)$, and calculate the perpendicular distance λ from $\mathbf{w}$ to $\psi(t)$. Figure 6 further illustrates the calculation of λ. With λ for all objects, we filter out objects with λ greater than a threshold η_{wall}. They are considered being too far away from the direction of the pending wall object. Then, among the remaining objects, we find the guide object with the least Euclidean distance to $\mathbf{\Lambda}$ as shown in Figure 6. Next, we use the guide object to derive an appropriate wall object. All dominant and subordinate objects are capable of deriving wall objects, depending on the learnt And-Or Graph [24] w.r.t. wall objects in the dataset. The derivation follows the process illustrated in Figure 6, where $\mathbf{\Lambda}$ is converted to the original positions of wall objects in the dataset.

7 EXPERIMENTS

7.1 Results and Setup

Several qualitative results generated by our framework are shown in Figure 7. To fully evaluate the proposed work, we develop a platform with sufficient functionalities compared with similar tools such as Planner5D[3] or Kujiale[4], as shown in Figure 8.

[3]https://planner5d.com/
[4]https://b.kujiale.com/

Figure 7: Our results on different room types. More generated scenes are included in the supplementary materials.

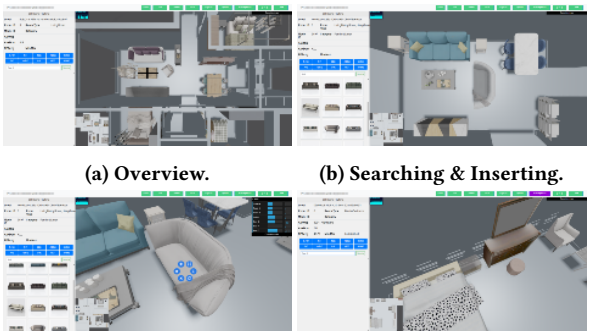


Figure 8: The open-source platform we implement for researches on 3D scenes. More details are included in Section D of the supplementary document.

Dataset. The dataset we are using is 3D-Front[5] [9, 10] with 70000+ rooms (layouts) and 9992 3D models (objects). Since 3D-Front does not have object categories such as "television", "laptop", "rug", etc., we further include additional models for a more diverse dataset in order to fully embody the effectiveness of our work. With the support of our platform, several end users are invited to create more layouts for learning subordinate priors, e.g., a double bed with its surroundings. As object categories do not directly reflect their usages, we relabelled object categories to suit our needs. An object is dominant, subordinate, or wall-related according to its category, e.g., dinning table is dom-object and nightstand is sub-object.

Implementation Details. The front-end rendering is based on Three.js[6], which is a popular rendering engine on top of WebGL. The back-end uses Numpy and Shapely for geometry operations such as room shape processing. We implement our own file systems for organizing scenes, 3D meshes, and priors. Our framework is developed on a desktop computer with GTX 2080ti GPU, 32GB memory, and AMD Ryzen 2700x CPU. The evaluation on computational efficiency is discussed in Section 7.4. Other practical concerns such as collision avoidance are discussed in Section C of the supplementary document.

[5]https://pages.tmall.com/wow/cab/tianchi/promotion/alibaba-3d-scene-dataset
[6]https://threejs.org/

7.2 User Satisfaction and Interaction

To verify how our framework benefits end users, we conducted a user study to measure the result quality and the system usability compared with traditional industrial solutions. We invited 37 participants (composed by office workers, university students, freelancers, etc.) to control cursors for scene synthesis. Given an empty room, participants were asked to generate two satisfied layouts (traditional approach vs. our approach) as quickly as possible.

The first layout generation follows the traditional object selection and arrangement process. The participants need to search an object (e.g., a desk or an office chair to suit the existing desk) and insert it into the scene, then optionally fine-tune its transformation. To conduct this experiment, we developed a platform for 3D scene manipulation as shown in Figure 8. In our platform, participants can search objects by typing in a search box or directly clicking an object name in a recommended list as shown in Figure 8b. After object selection, the object moves following the cursor until object insertion, which can be cancelled by a right click. We implement two transformation interfaces similar to Planner5d and Kujiale for object manipulation as shown in Figure 8c. First, by clicking an existing object in the scene we allow positioning, lifting, rotating and re-scaling the object based on the mouse cursor. Second, object transformation can be explicitly configured using a parameter panel

Table 1: User Satisfaction and Interactive Efficiency.

Measurement	Traditional	Ours
Interactive Satisfaction	3.51 (1.06)	3.52 (1.09)
Result Satisfaction	3.74 (0.96)	3.75 (0.83)
Time Consumption	835.21 (523.05)	437.6 (301.78)

(top-left). For generating the second layout, the participants were asked to use the proposed framework, as shown in Figure 8d.

Before the experiment started, a detailed manual was provided to the participants to tell them how to use the platform. We also showed them several well-designed scenes from the dataset as a standard in advance to avoid generating low-quality scenes. The initial empty rooms were selected with area greater than $25m^2$ to enable sufficient space for layout design. One technical staff was stood by in case of any technical question during the experiment.

Once a participant finished, we asked them to mark their overall satisfaction with the results. Likert-scale is adopted from 0: totally inaesthetic and implausible to 5: very aesthetic and plausible. We also asked them to mark degrees of convenience about the entire processes of interactions, ranging from 0: the interaction is poor and inconvenient to 5: the interaction is great and convenient. We also recorded the time consumed for generating each layout. The results are summarized in Table 1, where each cell includes an average value and a standard deviation in a bracket. It can be seen that our method significantly reduces the time consumption of arranging a scene while keeping both result quality and system usability.

According to our experiments, our framework alleviates the time consumption from the following factors: 1) search objects; 2) re-scale models for better fitness; 3) fail to remember accurate name of furniture, e.g., ottoman (in other languages); 4) spend time for transforming objects, e.g., place chairs in plausible positions and rotate them facing the dinning table.

7.3 Aesthetics and Plausibility

We next measure the aesthetics and plausibility attained by our framework by comparing with three existing scene synthesis techniques. **PlanIT** [27] is a state-of-art framework for 3D scene synthesis. **3D-Front** [9] presents the dataset used by our work. The **Geometry-Based Approach (GBA)** [39] is a recent method for generating room layouts. For PlanIT, we train it on 3D-Front and gather their synthesized results. For 3D-Front, we randomly pick scenes from the dataset. Note that rooms from 3D-Front are mainly designed by human and partially synthesized, e.g, using [29] to refine the object arrangements. GBA re-arranges existing scenes in 3D-Front without object selection. For our framework, we let computer automatically control the cursor and its movement, and insert objects accordingly. Consequently, all four methods take the same input and generate scenes respectively as shown in Figure 9.

Qualitatively, 3D-Front scenes often have leftover space such as Figure 10a, i.e., a room contains only a few objects but is suitable to have more objects. PlanIT [27] yields unexpected results between dom-objects and sub-objects as shown in Figure 10b, where it is also possible to generate physically implausible scenes. GBA [39] does not consider room capacity as shown in Figure 10c, e.g., a group of objects led by a dom-object may be too large to be put or the group occupies the entire room. In contrast, on the basis of physical plausibility, objects are harmoniously arranged among each other and rooms are compatibly filled according to their capabilities.

(a) 3D-Front [9]. (b) PlanIT [27]. (c) GBA [39]. (d) Ours.

Figure 9: Several results of the three baselines and ours given the same room shape. More qualitative comparisons can be found in Section E of the supplementary document.

(a) 3D-Front. (b) PlanIT. (c) GBA.

Figure 10: Several failure cases of the three baselines. 10a: the room has leftover space. 10b: relative transformations between the dom-objects and the sub-objects are unexpected. 10c: without considering room capacity, the group of objects are too big to be placed.

Quantitatively, another 50 participants were invited to evaluate the general aesthetics and plausibility online. We developed a web-based platform for collecting questionnaires as shown in Figure 11. Participants were asked to answer a series of questions. Each question contains 4 scenes that were synthesized by the above 4 methods given the same room. Participants were asked to compare the results and mark them separately from 0 to 5, where 0 denotes

Figure 11: The questionnaire platform for conducting the user study in Section 7.3. For each question, users respectively mark the presented scenes (LEFT), whereas they can zoom in each scene for better perceptions (RIGHT).

Table 2: Aesthetics and Plausibility.

Methods	3D-Front	PlanIT	GBA	Ours
Master Bedroom	3.327	2.103	2.437	3.443
Second Bedroom	3.577	1.33	2.62	3.28
Kids Room	3.653	1.82	2.373	3.37
Living-Dinning Room	3.163	2.117	2.003	3.603
Living Room	3.46	2.07	2.393	3.623
Dinning Room	3.237	1.85	2.443	3.427
Total	3.403	1.882	2.378	3.458

"very poorly generated" and 5 denotes "perfectly plausible and aesthetic". Before they started, they were explained how to use the questionnaire system, and an electronic guidance was available during the session. We chose 6 types of rooms as listed in Table 2. For each room type, 6 rooms were used to show the results. For each question, an empty room was randomly selected and the results of the 4 methods were rendered respectively.

The user feedbacks are shown in Table 2 according to room types. Our work outperforms PlanIT and GBA, and is on par with 3D-Front in terms of aesthetics and plausibility. Compared with human-guided scenes, our work is competitive for most room types and achieves slightly higher grades in general.

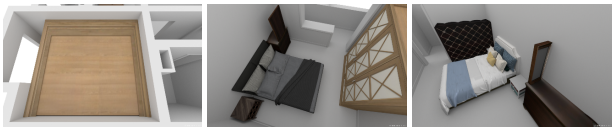
7.4 Efficiency

To evaluate the efficiency, we run our framework on PCs with different hardware configurations and inspect the frame rate in the front-end. A configuration considers GPU, CPU and RAM. Frame rate is measured in "frames per second" (FPS), including its average, minimum, and maximum rates.

This experiment is conducted in the most intensive situation of our framework, i.e., as shown in Figure 2, when a room is with several dom-objects inserted and fails to match sub-priors, our framework would then match the remaining priors of dominant objects, thus execute two workflows with two sets of priors. For doing so, we create a scene with several existing dom-objects, dom-priors and sub-priors. The cursor is moving rapidly within the scene to make the computational power as fully loaded as possible. The average, minimum, and maximum FPS are then calculated through the entire process. Table 3 shows the efficiency results. Column "Frame per Seconds" (FPS) writes averaged FPS achieved on a particular configuration. The following brackets show the minimum and maximum respectively. As a result, our framework is capable of smoothly operating on PCs with relatively more recent hardware configurations.

Table 3: Performances of framework on various devices.

GPU	RAM (MHz)	Frames per Second
HD Graphics 620	8GB (931.1)	28.78 (22.32 – 37.05)
Quadro FX 5800	32GB (931.1)	55.20 (48.86 – 59.81)
GTX 970	32GB (931.1)	122.10 (109.14 – 133.52)
GTX 1060	32GB (931.1)	143.11 (138.35 – 144.00+)
RTX 2080ti	32GB (931.1)	144.00+



(a) Weird Scale. (b) Orientation. (c) Distortion.

Figure 12: Three representative cases that are abnormal in the dataset. 12a: an extremely large wardrobe occupies a whole room. 12b: a dressing table faces in a wrong direction. 12c: a dressing table is distorted weirdly against the wall.

8 CONCLUSION AND FUTURE WORK

In this paper, we proposed and verified a framework of synthesizing 3D scenes involving and simulating human interactions. Our method is flexible in a sense that both automatic generation and user interaction are allowed. Through the experiments, we show our method is effective and outperforms the competing methods. We hope this work could advance future researches relying upon 3D virtual scenes.

Our work still has limitations and leaves room to be improved in the future. First, the way to interact with 3D scenes is empirically set according to our own experiments. In practice, not all users are accustomed to our current interaction setting. For example, one participant advises us to swap instances using mouse wheel instead of mouse click. As such, what are the natural user interactions are to be explored in the future.

As a data-driven approach, our work also suffers from several weird cases originated from the dataset. First, some objects are mislabelled in categories. Second, few objects are scaled too much, such as the wardrobe shown in Figure 12a, which influences the priors learnt by RDM. Third, few objects are wrongly rotated as shown in Figure 12b, where a dressing table is facing the wall. Similarly, the dressing table is highly distorted in Figure 12c. Our work also assumes that objects are centered at the origin in their local coordinate system, and are facing the direction of Z axis.

We also consulted several professional interior designers. They suggested that the existing state-of-the-art and our work still focus on "soft decorations", which refers to objects that are easy to move. In contrast, practical applications do require "hard decorations", e.g., holes on walls, pipelines, wires, furred ceilings, etc. In addition, "soft decorations" also favor style compatibility among objects, which can be achieved by a post-processing step such as [17].

ACKNOWLEDGMENTS

This work was supported by the National Key Technology R&D Program (Project Number 2017YFB1002604), the National Natural Science Foundation of China (Project Numbers 61772298, 61832016), Research Grant of Beijing Higher Institution Engineering Research Center, and Tsinghua–Tencent Joint Laboratory for Internet Innovation Technology. Yong-Liang Yang was partly supported by RCUK grant CAMERA (EP/M023281/1, EP/T014865/1), and a gift from Adobe.

REFERENCES

[1] autodesk.com. 2020. Autodesk Revit. Retrieved Dec 8, 2020 from https://www.autodesk.com/

[2] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. 2015. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289* (2015).

[3] Kang Chen, Yukun Lai, Yu-Xin Wu, Ralph Robert Martin, and Shi-Min Hu. 2014. Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information. *ACM Transactions on Graphics* 33, 6 (2014).

[4] Kang Chen, Yu-Kun Lai, and Shi-Min Hu. 2015. 3D indoor scene modeling from RGB-D data: a survey. *Computational Visual Media* 1, 4 (2015), 267–278.

[5] Francis DK Ching and Corky Binggeli. 2018. *Interior design illustrated*. John Wiley & Sons.

[6] ea.com. 2020. The Sims 4. Retrieved Dec 8, 2020 from https://www.ea.com/zh-cn/games/the-sims/the-sims-4

[7] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 135.

[8] Matthew Fisher, Manolis Savva, Yangyan Li, Pat Hanrahan, and Matthias Nießner. 2015. Activity-centric scene synthesis for functional 3D scene modeling. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 179.

[9] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Cao Li, Qixun Zeng, Chengyue Sun, Yiyun Fei, Yu Zheng, Ying Li, Yi Liu, Peng Liu, Lin Ma, Le Weng, Xiaohang Hu, Xin Ma, Qian Qian, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 2020. 3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics. *arXiv preprint arXiv:2011.09127* (2020).

[10] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 2020. 3D-FUTURE: 3D Furniture shape with TextURE. *arXiv preprint arXiv:2009.09633* (2020).

[11] Qiang Fu, Xiaowu Chen, Xiaotian Wang, Sijia Wen, Bin Zhou, and Hongbo Fu. 2017. Adaptive synthesis of indoor scenes via activity-associated object relation graphs. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

[12] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. 2016. Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE CVPR*. 4077–4085.

[13] kujiale.com. 2020. Kujiale. Retrieved Dec 8, 2020 from https://www.kujiale.com/

[14] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. 2019. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)* 38, 2 (2019), 1–16.

[15] Wanwan Li, Javier Talavera, Amilcar Gomez Samayoa, Jyh-Ming Lien, and Lap-Fai Yu. 2020. Automatic Synthesis of Virtual Wheelchair Training Scenarios. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 539–547.

[16] Yuan Liang, Song-Hai Zhang, and Ralph Robert Martin. 2017. Automatic data-driven room design generation. In *International Workshop on Next Generation Computer Animation Techniques*. Springer, 133–148.

[17] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. 2015. Style compatibility for 3D furniture models. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–9.

[18] Hong-Li Lu. 2010. *Residential Interior Design*. Liaoning Science and Technology Publishing House.

[19] Andrew Luo, Zhoutong Zhang, Jiajun Wu, and Joshua B Tenenbaum. 2020. End-to-End Optimization of Scene Layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3754–3763.

[20] Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. 2018. Language-driven synthesis of 3D scenes from scene databases. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 212.

[21] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive furniture layout using interior design guidelines. In *ACM transactions on graphics (TOG)*, Vol. 30. ACM, 87.

[22] Maureen Mitton and Courtney Nystuen. 2016. *Residential interior design: A guide to planning spaces*. John Wiley & Sons.

[23] planner5d.com. 2020. Planner5d. Retrieved Dec 8, 2020 from https://planner5d.com/

[24] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. 2018. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5899–5908.

[25] Manolis Savva, Angel X Chang, and Maneesh Agrawala. 2017. Scenesuggest: Context-driven 3d scene design. *arXiv preprint arXiv:1703.00061* (2017).

[26] Anna Bonarou Vasiliki Asaroglou. 2013. *Furniture arrangement: in Residential spaces*. CreateSpace Independent Publishing Platform.

[27] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. 2019. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 132.

[28] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. 2018. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 70.

[29] Tomer Weiss, Alan Litteneker, Noah Duncan, Masaki Nakada, Chenfanfu Jiang, Lap-Fai Yu, and Demetri Terzopoulos. 2018. Fast and scalable position-based layout synthesis. *arXiv preprint arXiv:1809.10526* (2018).

[30] Guoming Xiong, Qiang Fu, Hongbo Fu, Bin Zhou, Guoliang Luo, and Zhigang Deng. 2020. Motion Planning for Convertible Indoor Scene Layout Design. *IEEE Transactions on Visualization and Computer Graphics* (2020).

[31] Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. 2013. Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 123.

[32] Meng Yan, Xuejin Chen, and Jie Zhou. 2017. An interactive system for efficient 3D furniture arrangement. In *Proceedings of the Computer Graphics International Conference*. 1–6.

[33] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D Goodman, and Pat Hanrahan. 2012. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 56.

[34] Lap-Fai Yu, Sai Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley Osher. 2011. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph.* 30, 4 (2011), 86.

[35] Lap-Fai Yu, Sai-Kit Yeung, and Demetri Terzopoulos. 2015. The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE transactions on visualization and computer graphics* 22, 2 (2015), 1138–1148.

[36] Suiyun Zhang, Zhizhong Han, and Hui Zhang. 2016. User guided 3D scene enrichment.. In *VRCAI*. 353–362.

[37] Song-Hai Zhang, Shao-Kui Zhang, Yuan Liang, and Peter Hall. 2019. A Survey of 3D Indoor Scene Synthesis. *Journal of Computer Science and Technology* 34, 3, Article 594 (2019), 14 pages. https://doi.org/10.1007/s11390-019-1929-5

[38] Song-Hai Zhang, Shao-Kui Zhang, Wei-Yu Xie, Cheng-Yang Luo, Yong-Liang Yang, and Hongbo Fu. 2021. Fast 3D Indoor Scene Synthesis by Learning Spatial Relation Priors of Objects. *IEEE Transactions on Visualization and Computer Graphics* (2021). https://doi.org/10.1109/TVCG.2021.3050143

[39] Shao-Kui Zhang, Wei-Yu Xie, and Song-Hai Zhang. 2021. Geometry-Based Layout Generation with Hyper-Relations AMONG Objects. *Graphical Models* (2021), 101104. https://doi.org/10.1016/j.gmod.2021.101104

[40] Tao-Kai Zheng. 2011. *Interior Design For Home*. Huazhong University of Science & Technology Press.

[41] Yang Zhou, Zachary While, and Evangelos Kalogerakis. 2019. Scenegraphnet: Neural message passing for 3d indoor scene augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7384–7392.

[42] Fan Zhu, Li Liu, Jin Xie, Fumin Shen, Ling Shao, and Yi Fang. 2018. Learning to synthesize 3d indoor scenes from monocular images. In *Proceedings of the 26th ACM international conference on Multimedia*. 501–509.